P-中心选址问题的一种降阶回溯算法 *

尚春剑，宁爱兵，彭大江，张惠珍
(上海理工大学 管理学院, 上海 200093)

摘　要：运筹学研究领域中的应急服务设施选址问题有许多求解模型，该文选取了 P-中心模型进行研究，首先研究了该问题的数学性质，并给出证明，利用这些数学性质能对问题进行降阶从而缩小问题的规模，然后在此基础上设计一个基于上界和下界的回溯算法来求解该问题，最后通过一个示例分析进一步阐述该算法的原理，并证明了该算法能在较短时间内求得问题的最优解。

关键词：设施选址问题；P-中心模型；降阶算法；上界；下界；回溯算法
中图分类号：TP　　doi: 10.19734/j.issn.1001-3695.2020.04.0057

Backtracking algorithm with reduction for p-center location problem

Shang Chunjian, Ning Aibing, Peng Dajiang, Zhang Huizhen
(*Business School*, *University of Shanghai for Science & Technology*, *Shanghai* 200093, *China*)

Abstract: There are various solution models for the emergency service facility location problem and this paper studies the P-center model. This paper proposed a backtracking algorithm with upper and lower bound, and the algorithm added a reduction algorithm by studying the mathematical properties. The algorithm can decrease the scale and the degree of complexity of the problem, so as to accelerate the execution speed. At the end of the paper, this paper illustrated an instance to elaborate this algorithm further.

Key words: facility location problem; p-center model; reduction algorithm; upper bound; lower bound; backtracking algorithm

0　引言

日常生活中的各类紧急突发事件给人们的安全和生活造成了巨大的影响，当灾害发生时，应急服务设施是提供救援支持、保障人民群众人身安全最基本最重要的设施，具有十分重要的作用。

多数应急设施选址问题是 NP 难题，有多种不同的模型来解决这类问题。针对应急设施选址问题的公平性要求，本文选取 P-中心模型[1]进行研究，它是离散位置模型的一种特殊类型，是指从 n 个备选的设施中最多选取 p 个设施，对于每一个应急点 e_i 均由选中的设施中距离其最近的设施服务，要求所有应急点与对应为其提供服务的设施之间的最大服务距离最小，即尽量减少所有应急点与它们的服务设施之间的最大距离。在这个模型中，设施没有容量约束。目前有许多学者致力于此选址模型的研究，有的学者结合图论知识对问题模型进行求解，例如 Hockbaum[2]提出一个时间复杂度为 $O(E\log E)$的图的顶点多中心问题的 2-opt 算法，Dyer[3]提出一个时间复杂度为 $O(np)$的 2-opt 启发式算法；还有的学者对于 P-中心问题采用了模糊集的理论进行求解，例如 Wen[4]构建设施选址模型时考虑到模糊需求，实现应急设施选址的完全覆盖；有的学者采用启发式搜索的方法求解，例如 Domschke 等人[5]采用二分法的思想求解出中心值等于 p 的 r 覆盖问题，朱燕等[6]结合层次分析法，采用贪婪取走启发式算法对 p-中心模型进行求解，应用于航空救援服务；Davidovi 等[7]采用蜂群算法求解 P-中心问题；许彦宸等[8]利用改进 K-Means 算法和重心法相结合的方式对 P-中值问题进行求解，黄书强等[9]提出一种加权距离连续 K 中心选址问题求解方法；还有些学者研究精确算法求解 P-中心问题，例如 Sandoval 等人[10]提出基于整数规划模型的精确算法，Contardo 等[11]引入了一种可伸缩的基于松弛的迭代算法设计一种精确算法求解 P-中心问题。

该问题的启发式算法和近似算法的优点是能够快速的得到可行解，不足之处在于容易陷入局部最优[12, 13]；该问题的精确算法优点是能得到最优解，不足之处在于不适合求解大规模问题，具有较高的时间复杂度[14]。针对上有算法的不足之处，本文设计了一个精确算法，首先研究 P-中心模型的数学性质，并给出相应的数学证明，利用这些数学性质能批量某些确定开设或不开设的设施，进而缩小问题的规模，最后设计一个基于上界和下界的回溯算法，求得问题最优解的同时能利用数学性质加快问题求解速度。

1　问题描述

本文用二分图 $G = (E, F)$来表示该问题，$E = \{e_i \mid i = 1, 2, \cdots, m\}$表示所有应急点的集合，用 e_i 代表第 i 个应急点，m 代表应急点的总个数；$F = \{f_j \mid j = 1, 2, \cdots, n\}$表示设施的集合，用 f_j 代表第 j 个设施，n 代表设施的个数。

本文引入一个额外的决策变量 Q 来表示所有应急点和相应为其提供服务的设施之间的最大距离。已知从设施中最多选取 p 个设施，同时给定每个设施与应急点的相应位置关系，且 $p \le n$；$d_{i,j}$ 表示第 i 个应急点 e_i 到第 j 个设施 f_j 的距离；设决策变量：

$$y_{i,j} = \begin{cases} 1, \text{应急点 } e_i \text{ 由设施 } f_j \text{ 提供服务} \\ 0, \text{应急点 } e_i \text{ 不由设施 } f_j \text{ 提供服务} \end{cases}$$

收稿日期：2020-04-10；**修回日期：**2020-05-25　**基金项目：**国家自然科学基金资助项目(71401106)；上海市一流学科建设资助项目(S1201YLXK)；上海市教委管理科学与工程高原学科建设项目(2018-2021)；高等学校博士学科点专项科研基金联合资助课题(20123120120005)

作者简介：尚春剑(1994-)，女，河南信阳人，博士研究生，主要研究方向为算法、系统工程(scancan1216@qq.com)；宁爱兵(1972-)，男，湖南邵东人，副教授，博士，主要研究方向为算法设计、系统工程；彭大江(1995-)，男，四川成都人，博士研究生，主要研究方向为算法、系统工程；张惠珍(1979-)，女，山西人，副教授，博士，主要研究方向为优化算法.

chinaXiv:202009.00091v1

ChinaXiv合作期刊

$$x_j=\begin{cases}1, \text{设施 } f_j \text{ 被选取}\\ 0, \text{设施 } f_j \text{不被选取}\end{cases}$$

则 P-中心选址问题具体模型如下:

$$\min \quad Q$$

$$\text{s.t.}\begin{cases}\sum_{j\in\{j|f_j\in F\}} y_{i,j}=1, \quad i=1,2,\ldots\ldots m & (1)\\ y_{i,j}-x_j\le 0, \quad i=1,2,\ldots\ldots m;\ j=1,2,\ldots\ldots n & (2)\\ \sum_{j\in\{j|f_j\in F\}} x_j\le p, & (3)\\ \sum_{j\in\{j|f_j\in F\}} d_{i,j}y_{i,j}-Q\le 0, \quad i=1,2,\ldots\ldots m & (4)\\ x_j\in\{0,1\}, \quad j=1,2,\ldots\ldots n & (5)\\ y_{i,j}\in\{0,1\}, \quad i=1,2,\ldots\ldots m;\ j=1,2,\ldots\ldots n & (6)\end{cases}$$

目标函数 Q 表明要最小化每一个需求点与它的服务设施之间的最大距离。

约束(1)表示有且仅有一个 $y_{i,j}$ 是非零的，即有且仅有一个设施可以为一个应急点服务,保证了任意应急点 e_i 都能找到一个设施为其服务，满足完全覆盖的要求。

约束(2)表示当 $x_j = 0$ 必有 $y_{i,j} = 0$，即只有被选中开设的设施才能为应急点服务。

约束(3)表示最多开设 p 个设施。

约束(4)表示 Q 必须大于第 i 个应急点与它的服务设施之间的距离，由于对每个应急点均有这个约束，即 Q 必须大于任一应急点与其服务设施的距离。

约束(5)和(6)表示变量的取值为 0 或 1。

该问题是 NP-Hard 问题，除非 P＝NP，否则该问题不存在多项式时间的精确算法[15]。

2　数学性质及子算法设计

2.1　数学符号

为了便于描述，本文采用以下数学符号来描述：

e_i，f_j：表示第 i 个应急点；f_j 表示第 j 个设施；

m，n：表示应急点的个数；n：表示设施的个数；

$E = \{e_i \mid i = 1, 2, \cdots, m\}$：表示所有应急点的集合；

$F = \{f_j \mid j = 1, 2, \cdots, n\}$：表示设施的集合；

p：表示最多选取 p 个设施($p \le n$)；

$d_{i,j}$：表示第 i 个应急点 e_i 到第 j 个设施 f_j 的距离；

$mini_d_1(i)$：表示应急点 e_i 到所有设施的最小距离；

$mini_n_1(i)$：表示所有设施中，设施 $mini_n_1(i)$到应急点 e_i 的距离最小；

$mini_d_2(i)$：表示应急点 e_i 到所有设施的次小距离；

$mini_n_2(i)$：表示所有设施中，设施 $mini_n_2(i)$到应急点 e_i 的距离次小；

$maxi_d_1(i)$：表示应急点 e_i 到所有设施的最大距离；

$minj_d_1(j)$：表示设施 f_j 到所有的应急点的最小距离；

$maxj_d_1(j)$：表示设施 f_j 到所有的应急点的最大距离；

$min_d_1(i,X)$：表示应急点 e_i 到集合 X 中所有设施中的最小距离；

$min_n_1(i,X)$：表示在集合 X 的所有设施中，设施 $min_n_1(i,X)$到应急点 e_i 的距离最小；

$min_d_2(i,X)$：表示应急点 e_i 到集合 X 中所有设施中的次小距离；

$min_n_2(i,X)$：表示在集合 X 的所有设施中，设施 $min_n_2(i,X)$ 到应急点 e_i 的距离次小；

F_1：一定开设的设施集合，若 F_1 中任一设施不开设则不能取得最优解，为全局变量；

F_0：一定不开设的设施集合，若 F_0 中任一设施开设则不能取得最优解，为全局变量；

$F_5 = F / F_1 / F_0$：暂时未确定是否开设的设施集合，为全局变量；

FF_1：F_5 中的设施在回溯算法分情况时假设开设的设施放入集合 FF_1 中，为全局变量；

FF_0：F_5 中的设施在回溯算法分情况时假设不开设的设施放入集合 FF_0 中，为全局变量；

$FF_5 = F_5 / FF_1 / FF_0$：回溯算法分情况时暂时未处理的设施放入集合 FF_5 中，为全局变量；

$best_q$：回溯算法在当前状态下已知最好的目标函数值，为全局变量；

F_{best}：算法在当前状态下已知最好的目标函数值对应的开设设施集合，$|F_{best}| \le p$，为全局变量；

u_q：算法在某一状态下的目标函数值上界，即 J_1 和 JJ_1 中设施开设时的上界，为局部变量；

cur_n：当前累计开设设施数，为局部变量；

cur_i：回溯算法的当前搜索层数，为局部变量；

opt_flag：若 $opt_flag = 1$，说明当前算法已取得最优解；若 $opt_flag = 0$，不能确定当前是否已取得最优解，为全局变量。

2.2　数学性质

性质 1　若存在一设施 f_j，假如设施 f_j 开设且此时的上界 u_q 小于当前的下界 $best_q$，则设施 f_j 一定不开设，其中上下界求解算法见后面 2 节的内容。

证明　上界 u_q 小于当前的下界 $best_q$，说明此时开设设施 f_j 不可能获得最优解，所以应该关闭设施 f_j。

性质 2　若存在某一设施 f_j，假如设施 f_j 不开设且此时上界 u_q 小于当前的下界 $best_q$，则设施 f_j 一定开设。

证明　上界 u_q 小于当前的下界 $best_q$，说明此时关闭设施 f_j 不可能获得最优解，所以应该开设设施 f_j。

性质 3　当 $p = 1$ 时，则此时开设 $maxj_d_1(j)$最小的设施即可。

证明　当 $p = 1$ 时，即此时只要选取 1 个设施，选 $maxj_d_1(j)$最小的设施能使目标函数最小。

性质 4　存在设施 f_j，f_h，若对于任意应急点 $e_i \in E$，有 $d_{i,j} \le d_{i,h}$，则设施 f_h 一定不开设，即将设施 f_h 放入集合 F_0 中。

证明　因为此时连接到设施 f_h 的应急点都可以连接到设施 f_j 上且目标函数相同或更好。

性质 5　存在设施 f_j，若其 $minj_d_1(j)$大于等于其他任一设施 f_h 的 $maxj_d_1(h)$，则设施 f_j 一定不开设，即将设施 f_j 放入集合 F_0 中。

证明　因为此时连接到设施 f_h 的应急点都可以连接到设施 f_j 上且目标函数相同或更好。

性质 6　存在两个应急点 e_i，e_k，若对于任意设施 $f_j \in F$，有 $d_{i,j} \ge d_{k,j}$，则可先删除应急点 e_k，算法最后再把应急点 e_i 连接到任一开设的设施都不影响目标函数值。

证明　由于对于任意设施 $f_j \in F$，有 $d_{i,j} \ge d_{k,j}$，因此把应急点 e_i 连接到任一开设的设施都不影响目标函数值。

性质 7　存在两个应急点 e_i，e_k，若 $mini_d_1(i) \ge maxi_d_1(k)$，则可先删除节点 e_k，算法最后再把应急点 e_k 连接到任一开设的设施都不影响目标函数值。

证明　由于必须服务应急点 e_i，因此目标函数一定大于等于 $mini_d_1(i)$，而应急点 e_k 到所有设施的最大距离小于等于 $mini_d_1(i)$，即应急点 e_k 到所有设施的最大距离一定小于等于目标函数，因此把应急点 e_k 连接到任一开设的设施都不影响目标函数值。

性质 8　存在设施 f_j，若其 $maxj_d_1(j)$小于等于其他任一个设施 f_h 的 $minj_d_1(h)$，则设施 f_j 必定开设，将其放入集合 F_1 中，且最优函数值即为 $maxj_d_1(j)$。

证明　此时所有应急点都连接到设施 f_j 即可取得最优解。

chinaXiv:202009.00091v1

ChinaXiv合作期刊

性质 9　若当前未被服务的应急点个数小于等于 p-$|F_1|$-$|FF_1|$，此时对于每一个未服务应急点 e_i，把应急点 e_i 连接到设施 $min_n_1(i, F - F_0 - FF_0)$即可，这样处理当前未服务应急点是最优的选择。

证明　这样操作开设的设施总个数小于等于 p 且当前未服务应急点都是连接到距离其最近的设施上，这样处理当前未服务应急点是最优的选择。

性质 10　将每一个应急点 i 都与距离其最近的设施相连接，若此时开设的设施总数量小于等于 p，则此时求得最优解。

证明　显然成立。

性质 11　将应急点 $mini_d_1(i)$值从大到小排序，然后依次将应急点 e_i 连接到设施 $mini_d_1(i)$，直到开设的设施数为 p 或者所有应急点都被服务为止；之后若还有应急点没有被服务，则把这些应急点连接到已经开设的 p 个设施中距离该应急点最近的设施上，若这些应急点连接的距离都小于或等于 $max(mini_d_1(i))$，则此时取得最优解，$opt_flag = 1$。

证明　此时目标函数值为下界 $max(mini_d_1(i))$，因此成立。

2.3　下界子算法

本文设计的下界子算法具体步骤如下：

a) 初始化，对任意 $e_i \in E$，$f_j \in F$，让每一个应急点均由所有设施中距离其最近的设施服务；

b) 判断当前设施开设个数。若当前设施开设个数小于等于 p，此时可求得最优解，最优函数值 $Q = max(mini_d_1(i))(e_i \in E)$，输出目标函数值，算法结束；若此时开设设施个数大于 p，此时令 $max(mini_d_1(i)) = b_0$，执行下一步；

c) 令 T 为前面开设的所有设施的集合，T 中设施的个数为 $count$；将 T 中每个设施 f_t 服务的应急点 e_{it} 都放入对应的集合 R_t 中；

d) 对 T 中的每一个设施 f_t，执行如下操作：若集合 R_t 中的任一应急点 e_i 都满足 $min_d_2(i, T) \le b_0$，则把设施 f_t 加入集合 M，也就说当设施 f_t 服务的任一应急点到设施集合 T 的次小距离值都比当前的 b_0 小，则把设施 f_t 加入集合 M；若集合 R_t 中的某一应急点 e_i 满足 $min_d_2(i, T) > b_0$，则把设施 f_t 加入集合 N，也就说当设施 f_t 服务的某一应急点到设施集合 T 的次小距离值比当前的 b_0 大，则把设施 f_t 加入集合 N；

e) 若 M 不为空，对 M 中的每一个设施 f_{tm}，执行如下操作：将设施 f_{tm} 对应集合中的 R_{tm} 中的所有应急点 e_{itm} 连接到其对应的 $min_n_2(i_{tm}, T)$上，并从集合 T 中删除对应的设施 f_{tm}，更新 $count$，更新每个应急点到集合 T 中各设施的最短与次短距离，若此时 $count \le p$，则此时求解得到最优解，$opt_flag = 1$，整个算法结束；若 $count > p$,执行步骤 f)；若 M 为空，执行步骤 f)；

f) 更新初始下界。对于集合 N 中每一个集合 R_t 的 $max(mini_d_2(i))$，将这些最大值 $max(mini_d_2(i))$中的最小值令为 b_t，若 $b_t > b_0$，则更新下界为 b_t；若 $b_t \le b_0$，则下界仍为 $b_0 = max(mini_d_1(i))$，算法结束。

2.4　上界子算法

事实上，任何一个可行解对应的目标函数值均能作为该问题的上界。本文先利用前面所介绍的数学性质，将问题进行降阶处理，之后执行如下的上界子算法求出上界：

a) 对于所有 $e_i \in E$，按照其 $mini_d_2(i)$的降序排列；排序之后，依此将应急点 e_i 连接到对应的设施 $mini_n_1(i)$上，逐一开设对应设施，直到设开设个数为 p 个或者所有应急点都被服务为止；将开设的设施放入集合 U 中，将还没有开设的的设施放入集合 V 中；

b) 若所有的应急点全部被集合 U 中的设施服务，则此时求解得到最优解，$opt_flag = 1$，整个算法结束；否则继续执行下一步；

c) 将所有还没有被服务的应急点 e_{iu} 与集合 U 中对应的设施 $min_n_1(i_u, U)$相连接；

d) 对于所有 $e_i \in E$，将 $max(min_d_1(i, U)) = u_0$ 作为上界，令 $max(min_d_1(i_u, U))$所对应的设施设为 f_h；

e) 若集合 V 中存在一个设施 f_j，若用设施 f_j 来替换开设的设施 f_h，即执行 $U = U \cup f_j - f_h$，$V = V \cup f_h - f_j$，之后如果对于所有 $e_i \in E$，$max(min_d_1(i, U)) < u_0$，则决定用设施 f_j 来替换开设的设施 f_h，跳到步骤 d)执行；若不存在这样的设施 f_j，则该子算法结束。

3　降阶回溯算法

本算法分为降阶算法和回溯算法两个部分，降阶算法通过结合数学性质进行降阶，缩小问题的规模；回溯法是采用深度优先法搜索解空间。回溯算法对设施集合 FF_5 中的每一个设施分为以下 2 种情况进行处理：a)若设施 f_j 开设，$FF_1 = FF_1 \cup \{f_j\}$，$FF_5 = FF_5 / \{f_j\}$，并搜索对应的左子树；b)若设施 f_j 不开设，$FF_0 = FF_0 \cup \{f_j\}$，$FF_5 = FF_5 / \{f_j\}$，并搜索对应的右子树。

3.1　降阶子算法

降阶子算法步骤如下：

a) 初始化 $opt_flag = 0$，$cur_i = 1$，$cur_n = 0$，$best_q = +\infty$；$u_q = 0$；

b) 结合前文的数学性质，若某个设施 f_j 一定开设，$F_1 = F_1 \cup \{f_j\}$，$F_5 = F_5 / \{f_j\}$；若某个设施 f_j 一定不开设，$F_0 = F_0 \cup \{f_j\}$，$F_5 = F_5 / \{f_j\}$。若执行以上操作后集合 F_5 有变化，则重新执行步骤 a)；

c) 根据前文的上界子算法对问题计算上界，此时开设情况赋给集合 F_{best}；对应的目标函数值赋给 $best_q$；

d) 调用下一节介绍的回溯子算法。

3.2　回溯子算法

本文回溯法采用深度优先法搜索问题解空间，从根节点出发遍历所有解空间，搜索至任一节点时，判断该节点的理论上界 $best_q$ 是否大于下界 b_q，若满足则继续向下深度优先搜索，否则进行剪枝，即跳过该节点及以下的子树，逐级向上回溯。

算法执行前先调用前面介绍的降阶子算法，若 $opt_flag = 1$，说明当前已取得最优解，算法结束；否则执行回溯子算法。为便于描述，对 F_5 中的设施重新编号，初始化 $FF_5 = F_5$，$FF_1 = \{\}$，$FF_0 = \{\}$，再调用递归回溯子程序 backtrack(1)。

回溯子程序 backtrack(cur_i)描述如下：

a) 若当前搜索层数 $cur_i > |F_5|$或者$|FF_5|=0$，说明搜索到达叶子节点，此时若$|F_1| + |FF_1| \le p$，则得到一个可行解。若对应的目标函数值 $Q < best_q$，则更新最优解，$best_q = Q$，$F_{best} = F_1 \cup FF_1$；结束该子程序；

b) 检查已开设设施 $F_1 \cup FF_1$ 是否已覆盖所有应急点且 $F_1 \cup FF_1 \le p$，若满足，则达到叶子节点，得到一个可行解。若对应的目标函数值 $Q < best_q$，则更新最优解，$opt_flag = 1$，$best_q = Q$，$F_{best} = F_1 \cup FF_1$；结束该子程序；

c) 在假设设施 f_{cur_i} 开设的情况下，计算下界 b_q。若$|FF_1| < p - |F_1|$且$|FF_5| \ge p - |F_1| - |FF_1|$，同时 $b_q < best_q$，则开设当前设施，执行 $FF_1 = FF_1 \cup \{f_{cur_i}\}$，$FF_5 = FF_5 / \{f_{cur_i}\}$，调用 backtrack($cur_i + 1$)进入左子树搜索；若上述条件不成立，则跳到 Step 6，此时左子树剪枝；

d) 若 $opt_flag = 1$，说明当前算法已取得最优解，结束该子程序；否则继续执行该子程序；

e) 算法跳到搜索树上一层前，恢复 FF_1 和 FF_5 到步骤 c)的初始状态，$FF_1 = FF_1 / \{f_{cur_i}\}$，$FF_5 = FF_5 + \{f_{cur_i}\}$；

f) 在假设设施 f_{cur_i} 不开设的情况下，计算下界 b_q。若

chinaXiv:202009.00091v1

ChinaXiv合作期刊

$|FF_1| < p - |F_1|$且$|FF_5| \geq p - |F_1| - |FF_1|$，同时 $b_q < best_q$，则不开设当前设施，执行 $FF_0 = FF_0 \cup \{f_{cur_i}\}$，$FF_5 = FF_5 / \{f_{cur_i}\}$，调用 backtrack($cur_i$ + 1)进入右子树搜索；若上述条件不成立，则结束该子程序，此时右子树剪枝。

g) 返回搜索树上一层前，恢复 FF_0 和 FF_5 到步骤 f)的初始状态，$FF_0 = FF_0 / \{f_{cur_i}\}$，$FF_5 = FF_5 + \{f_{cur_i}\}$；

算法结束后，当前的最优目标函数值 $best_q$ 和对应开设的设施 F_{best} 就是整个问题的一个最优解。

3.3 算法时间复杂度分析

本文研究的问题规模即设施个数为 n，该算法在搜索空间树中最多产生 2^n 个叶子节点，在降阶子算法被调用后，问题规模减少为$|F_5|$，令 $k = |F_5|$，因此算法在最坏情况下的时间复杂度为 $O(2^k)$。

针对应急选址问题许多学者提出不同算法，这些算法主要分为精确算法、近似算法和启发式算法。近似算法与启发式算法的优点是求解速度快，不足之处在于一般无法得到问题的最优解。本文的精确算法相对于一般的精确算法而言，通过研究其数学性质，对问题进行降阶，缩小问题的解空间，调用回溯算法前对搜索树进行合理剪枝，只对部分解子树搜索，加快了求解速度。

4 示例分析及实例对比

4.1 示例分析

为了更清晰的说明本文算法，下面给出一个示例对算法执行的整个过程进行说明。

示例 1 如图 1 所示，现有 6 个设施 j，5 个应急点 i，从中最多选出 $p = 3$ 个设施，使得最大需求距离最小，表 1 给出了每个应急点与设施之间的距离。

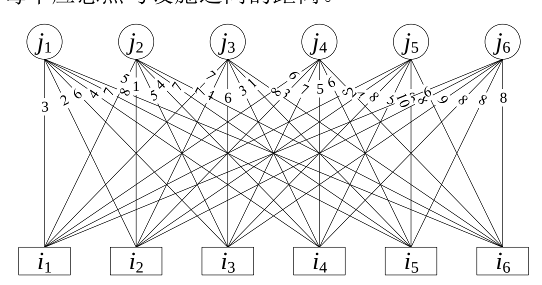


图 1 应急点与设施二分图

Fig. 1 Bipartite graph of emergency points and facilities

表 1 应急点与设施之间的距离

Tab. 1 The distance between the emergency point and the facility

i	j					
	j_1	j_2	j_3	j_4	j_5	j_6
i_1	3	5	7	6	8	10
i_2	2	1	4	3	4	8
i_3	6	5	6	7	8	9
i_4	4	4	3	5	5	8
i_5	7	7	1	6	6	8
i_6	8	7	8	2	6	8

对示例 1 的分析过程可描述如下：

a)根据性质 4，设施 j_5 相对于施 j_4 严格劣势，故从原二分图中删除设施 j_5 并且删除边$\{i_1j_5$，i_2j_5，i_3j_5，i_4j_5，i_5j_5，$i_6j_5\}$；

b)根据性质 4 或者性质 5，设施 j_6 可以删除，故从原二分图中删除设施 j_6 并且删除边$\{i_1j_6$，i_2j_6，i_3j_6，i_4j_6，i_5j_6，$i_6j_6\}$；

c)根据性质 6，应急点 i_2 相对于 i_1 可以不考虑，故将应急点 i_2 删除，从原二分图中删除应急点 i_2 并且删除边$\{i_2j_1$，i_2j_2，i_2j_3，i_2j_4，i_2j_5，$i_2j_6\}$；

d)根据性质 7，应急点 i_4 相对于 i_3 可以不考虑(因为之前设施 j_6 已被删除)，故将应急点 i_4 删除，从原二分图中删除应急点 i_4，并且删除边$\{i_4j_1$，i_4j_2，i_4j_3，i_4j_4，i_4j_5，$i_4j_6\}$；

e)经过降阶处理后，更新问题规模如图 2。

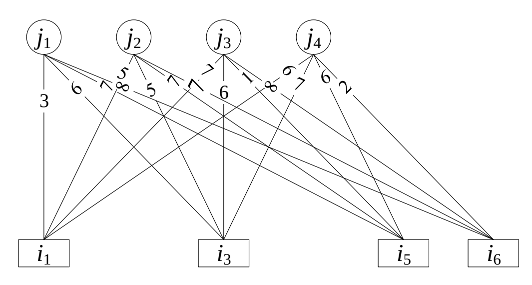


图 2 问题降阶处理后的二分图

Fig. 2 Bipartite graph of the problem after order reduction processing

f)此时将当前剩余应急点与剩余设施集合中距离其最近的设施相连，计算当前开设设施数得 $4 > p$，此时计算出下界 $b = 5$。

g)执行上界子算法，计算出上界 $u = 6$。

h)执行回溯算法，其执行过程如图 3 所示的二叉树。

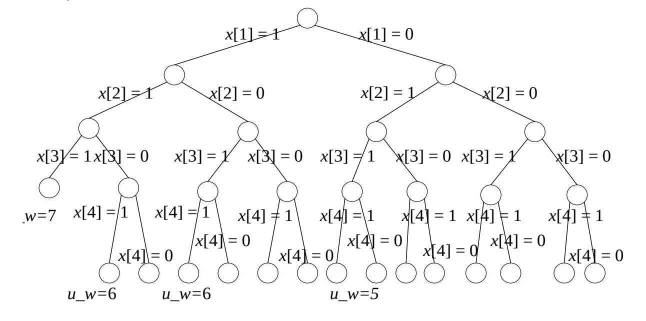


图 3 解空间二叉树

Fig. 3 Solution space binary tree

降阶后的解空间二叉树如图所示，通过上述算法得到该问题的最优解集 $F_{best} = \{j_2, j_3, j_4\}$，最优目标函数值 $best_q = 5$，算法结束。

4.2 实验结果对比与分析

本文实验在 CPU 为 Core i7-7500 @2.70GHz，操作系统为 64 位 Windows10 的环境下进行。采用 Benchmark Library 中提供部分基准数据进行实验，选取 The P-median Problem 实例中部分算例，$m = n = 128$，$p = 16$，该数据中服务距离取值为(0, 4)，采用 Python 编程实现本文精确算法(BA)和蚁群算法(ACO)，对相同数据进行求解，并对两种算法的结果进行对比分析，实验结果记录了问题求解结果和程序运行时间，如表 2 所示。

表 2 实验结果对比

Tab. 2 Comparison of experimental results

pro	result		Running time/s	
	BA	ACO	BA	ACO
Pmp334	3	4	25.879467	1.212099
Pmp434	4	4	25.254210	1.123443
Pmp534	3	4	27.854497	1.032928
Pmp634	3	4	25.309252	1.150121
Pmp734	2	4	26.830491	1.230132

从示例 1 可以清楚地看到，利用数学性质能降低问题的规模，加快算法的执行速度；从上述实验结果可得，实验结果与理论分析的结论一致：蚁群算法是启发式算法，计算速度比精确算法快，而本文设计的算法是精确算法,每次都能求得最优解，因此在上述实验中本文算法求出的结果稳定性要优于启发式算法类型的蚁群算法。

针对于选址问题设计算法的国内外研究一般分为启发式

chinaXiv:202009.00091v1

ChinaXiv合作期刊

算法、近似算法和精确算法，本文提出的降阶回溯算法是精确算法，该算法利用了数学性质、上界及下界来加快算法的运行速度，对比于启发式算法和近似算法，本文设计的精确算法保证每次都能得到正确稳定的全局最优解，而启发式算法容易陷入局部最优，近似算法也只能以一定的相似比接近最优解，一般无法得到问题的最优解；该问题的精确算法相关研究内容较少，与一般的精确算法相比最终的解是一样的，但是一般精确算法都是对所有节点进行分析，时间复杂度比较高，本文利用了问题的数学性质，结合设计的上下界子算法，能够批量确定某些开设或不开设的设施，降低问题的规模，只需分析部分节点，在一定程度上降低了精确算法处理大规模问题的困难，因此本文设计的算法比一般的精确算法速度快。

5 结束语

本文研究应急设施选址问题中的 P-中心模型，针对启发式算法无法得到问题最优解和精确算法处理大规模问题具有较高时间复杂度的劣势，本文设计了一种能快速降低问题规模同时能得到精确解的降阶回溯精确算法，该算法的设计目的在于保证获得全局最优解的情况下，尽可能减少求解时间。文章通过研究 P-中心选址问题的数学性质并给出相应证明，这些数学性质结合所设计的上下界子算法，在搜索过程中能批量确定某些开设或不开设的设施，在回溯过程中进行大量剪枝，从而降低问题求解规模，提高了算法的执行速度；同时，这些数学性质也可以应用于其他算法中。通过理论分析及示例 1 可以看出，采用本文的算法求解 P-中心选址问题时不仅能求出精确解，同时利用数学性质和上下子算法能降低问题的规模，加快问题求解速度。

参考文献：

[1] Hakimi S L. Optimum locations of switching centers and the absolute centers and medians of a graph [J]. Operations Research, 1964, 12 (3): 450-459.

[2] HOCHBAUM D S, SHMOYS D B. A best possible heuristic for the k-center problem [J]. Mathematics of Operations Research, 1985, 10 (2): 180-184.

[3] DYER M E, FRIEZE A M. A simple heuristic for the p-center problem [J]. Operations Research Letters, 1985, 3 (6): 285-288.

[4] WEN M, KANG R. Some optimal models for facility location-allocation problem with random fuzzy demands [J]. Applied Soft Computing, 2011, 11 (1): 1202-1207.

[5] Bozkaya B, Tansel B. A spanning tree approach to the absolute p-center problem [J]. Location Science, 1998, 6 (1): 83-107.

[6] 朱燕，邵荃，贾萌. 通用航空应急救援点布局方法研究 [J]. 河南科学, 2015 (2): 265-270. (Zhu Yan, Shao Quan, Jia Meng. Research on the layout of general aviation rescue point [J]. Henan Science, 2015 (2): 265-270.)

[7] Davidovi T, Ramljak D, Elmi M, *et al.* Bee colony optimization for the p-center problem [J]. Computers & Operations Research, 2011, 38 (10): 1367-1376.

[8] 许彦宸，戴韬. 基于 K-Means 算法和重心法求解多配送中心选址问题 [J]. 物流技术, 2019, 38 (06): 69-73. (Xu Yanchen, Dai Tao. Solving multiple distribution center location allocation problem using K-Means algorithm and center of gravity method [J]. Logistics Technology, 2019, 38 (06): 69-73.)

[9] 黄书强，江秀美，范人胜. 一种加权距离连续 K 中心选址问题求解方法 [J]. 小型微型计算机系统, 2020, 41 (02): 310-315. (Huang Shuqiang, Jiang Xiumei, Fan Rensheng. Method for weighted distance continuous K-center facility location problem [J]. Journal of Chinese Computer Systems, 2020, 41 (02): 310-315.)

[10] Sandoval MG, Diaz JA, Rios-Mercado RZ. An improved exact algorithm for a territory design problem with p-center-based dispersion minimization [J]. Expert Systems with Applications, 2020, 146 (1): 113150.

[11] Contardo C, Iori M, Kramer R. A scalable exact algorithm for the vertex p-center problem [J]. Computers & Operations Research, 2019, 103 (MAR): 211-220.

[12] 周建杰. 图上的路选址问题与连通 p-中心和 p-中位问题 [D]. 上海: 上海大学, 2016. (Zhou Jianjie. The problem of road location on graph and the problem of connecting p-center and p-median [D]. Shanghai: Shanghai University, 2016.)

[13] Perez-Pelo S, Sanchez-Oro J, Lopez-Sanchez AD. A multi-objective parallel iterated greedy for solving the p-center and p-dispersion problem, Electronics, 2019, 8 (12): 1140.

[14] Oezsoy F A, Pinar M C. An exact algorithm for the capacitated vertex p-center problem [J]. Computers & Operations Research, 2006, 33 (5) .

[15] Michael R Garey, David S Johnson, Computers and intractability: A guide to the theory of NP-completeness [J]. Bulletin (New Series) of the American Mathematical Society, 1980, 3 (2): 898-904.

chinaXiv:202009.00091v1